4/16/2004

FIG. 1

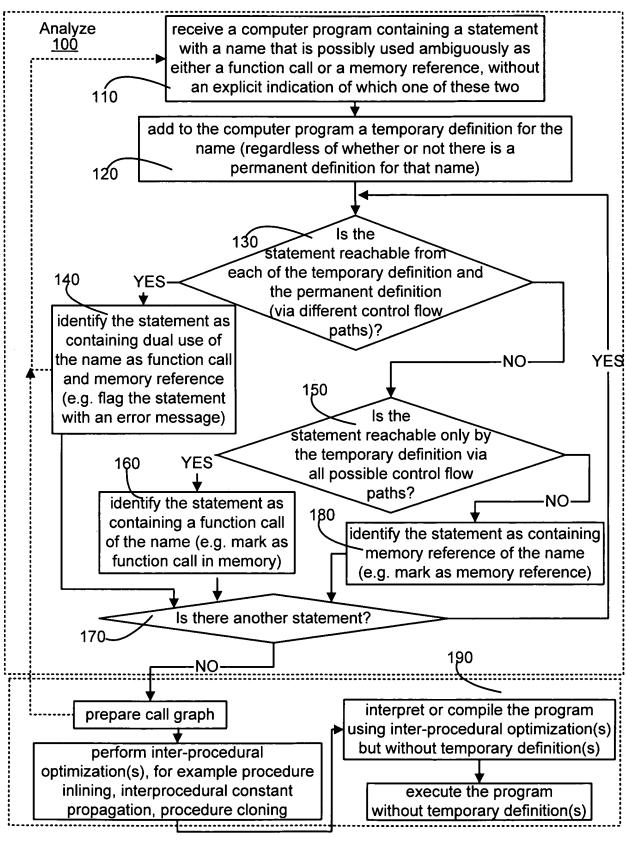
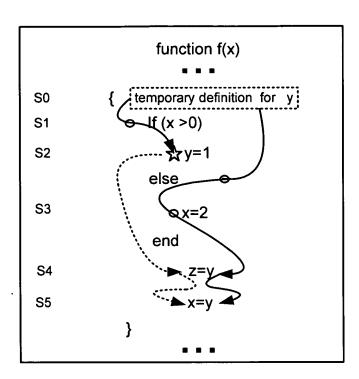


FIG. 2A



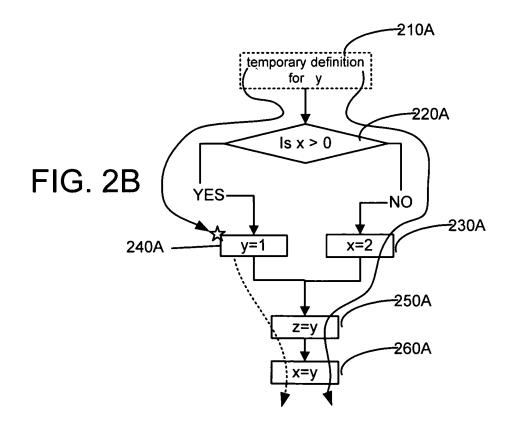
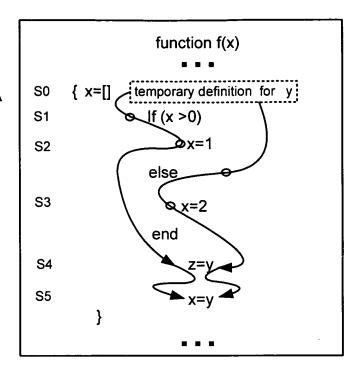


FIG. 3A



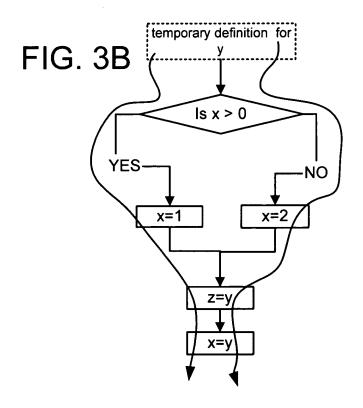
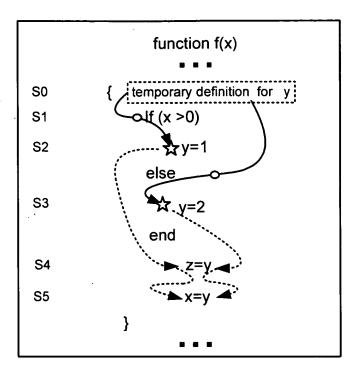
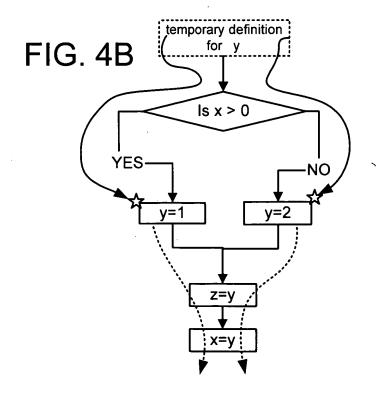
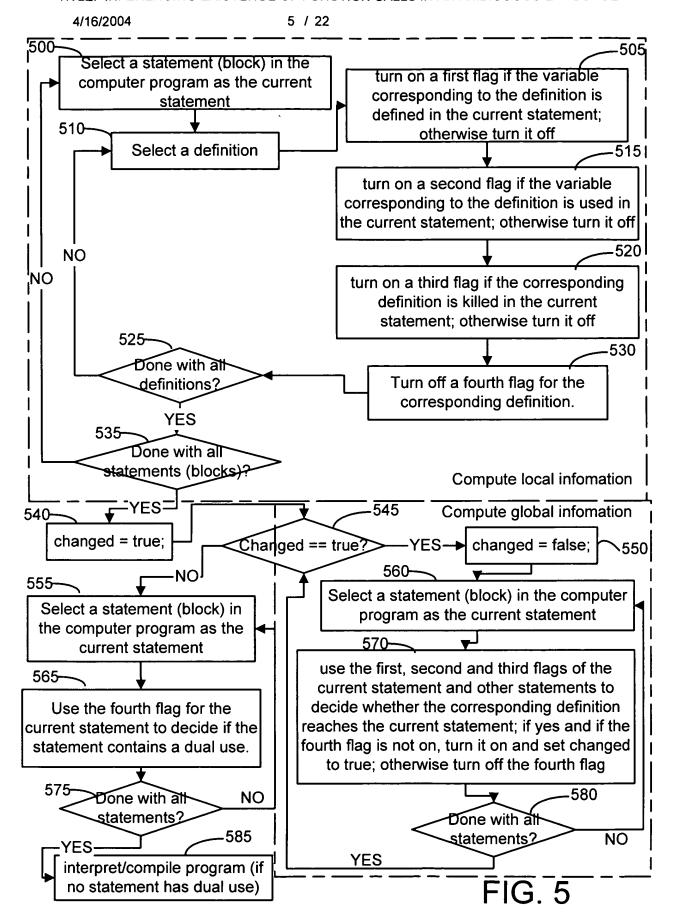


FIG. 4A







TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE

4/16/2004

6 / 22

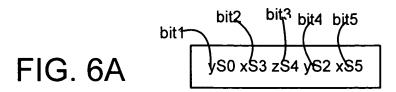


FIG. 6B

	DEFINES	KILLS	REACHES	USES
S0	10000	10010	00000	00000
<b>S1</b>	00000	00000	10000	01001
S2	00010	10010	10000	00000
<b>S</b> 3	01000	01001	10000	00000
S4	00100	00100	11010	10010
S5	00001	01001	11110	10010

FIG. 6D

	DEFINES	KILLS	REACHES	USES
S0	10000	10000	00000	00000
<b>S1</b>	00000	00000	10000	01011
S2	00010	01011	10000	00000
S3	01000	01011	10000	00000
<b>S4</b>	00100	00100	11010	10000
<b>S5</b>	00001	01011	11110	10000

TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE

4/16/2004

7 / 22

FIG. 6E

bit2 bit3 bit4 bit5 bit1 yS0 yS3 zS4 yS2 xS5

FIG. 6F

	DEFINES	KILLS	REACHES	USES
S0	10000	11010	00000	00000
<b>S1</b>	00000	00000	10000	00001
S2	00010	11010	10000	00000
<b>S3</b>	01000	11010	10000	00000
S4	00100	00100	01010	11010
S5	00001	00001	01110	11010

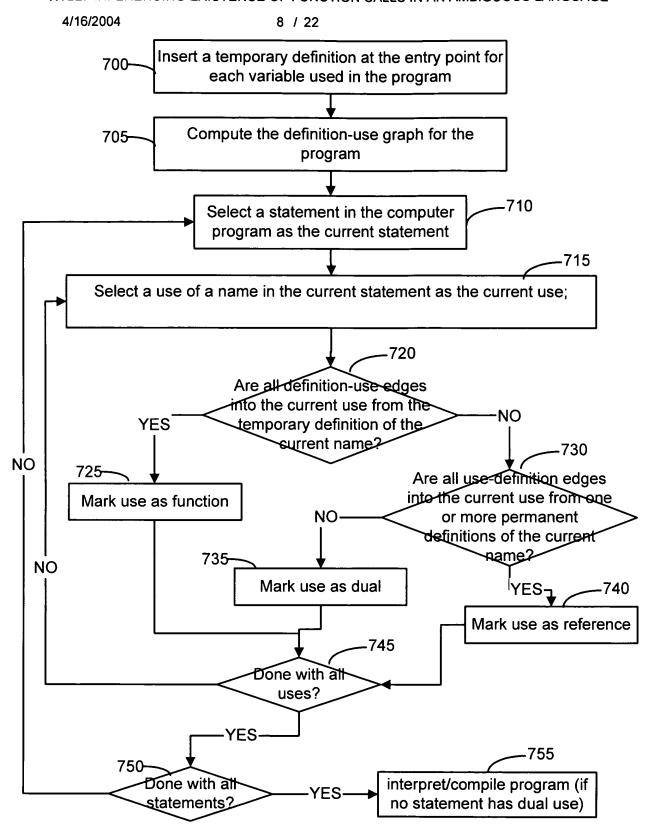


FIG. 7A

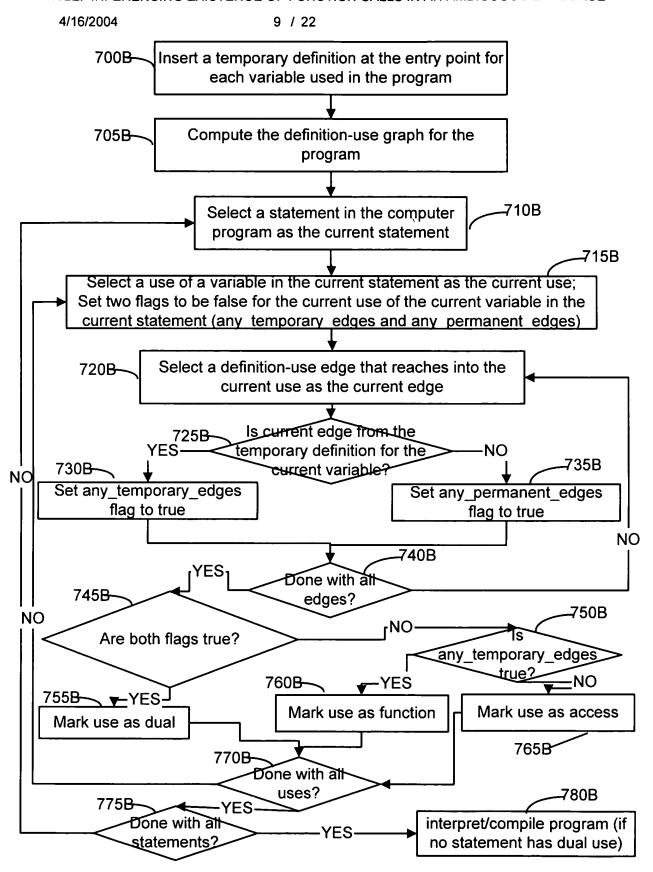


FIG. 7B

TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE

4/16/2004

FIG. 8A

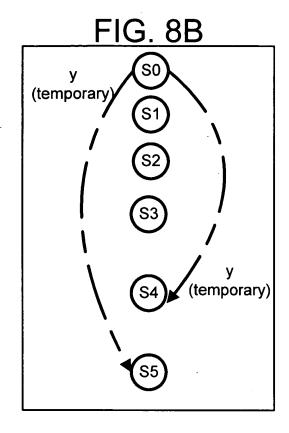
(temporary)

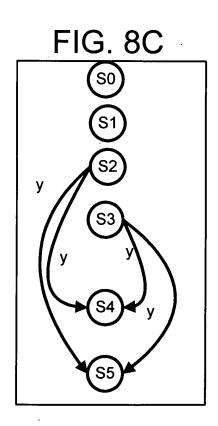
(s2)

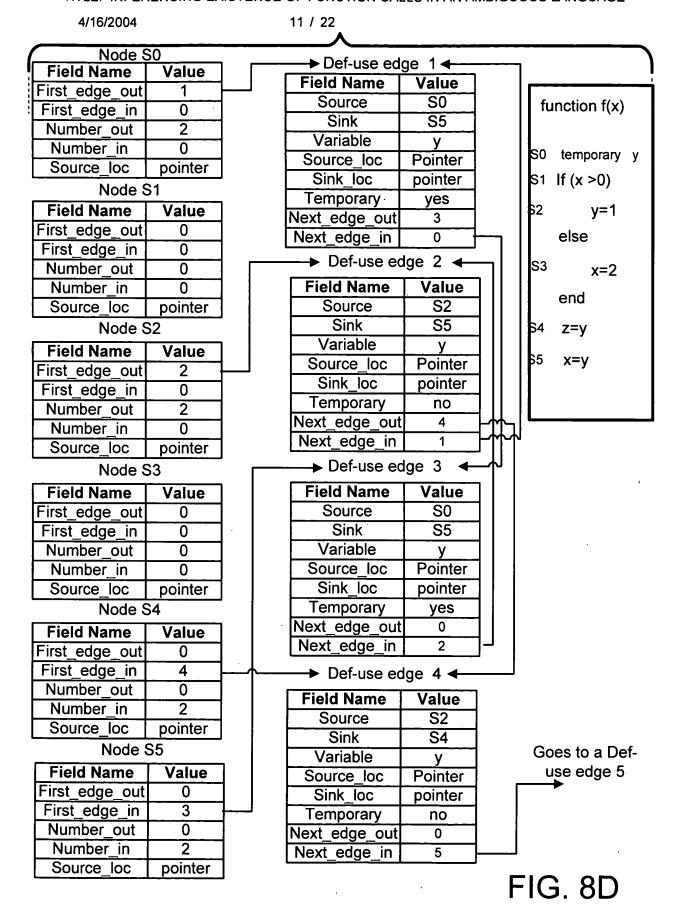
(s3)

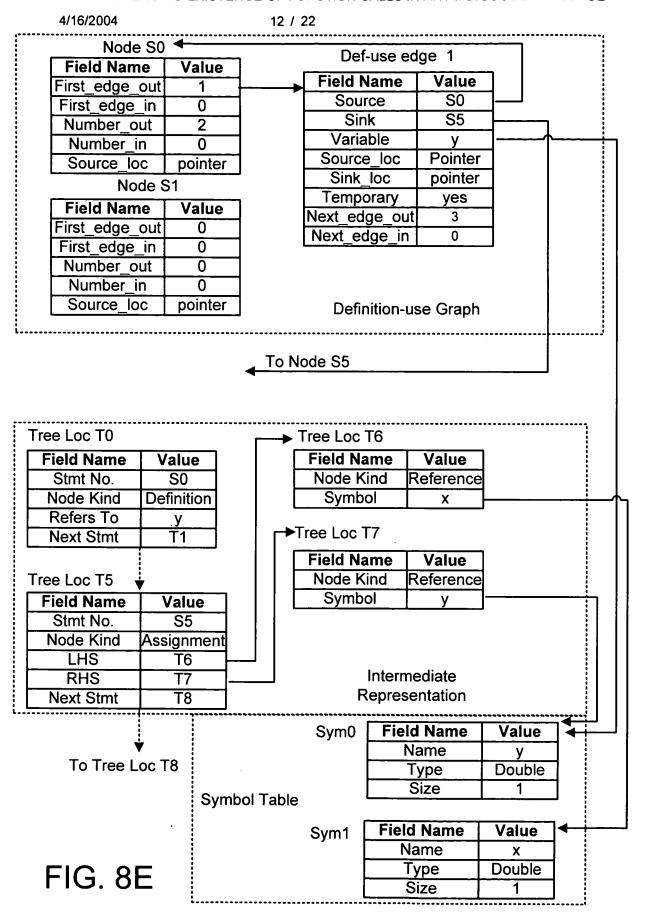
(s4)

(s5)









# ATTORNEY DOCKET: CAT003 US TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE 4/16/2004 13 / 22

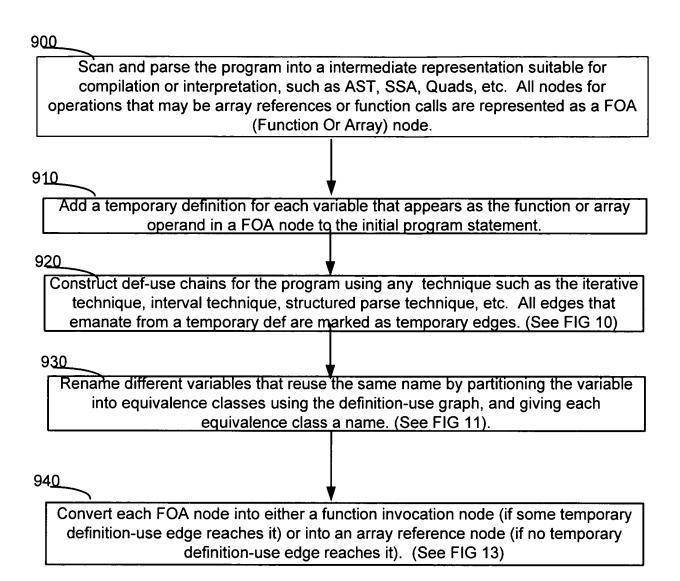


FIG. 9

TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE

4/16/2004

14 / 22

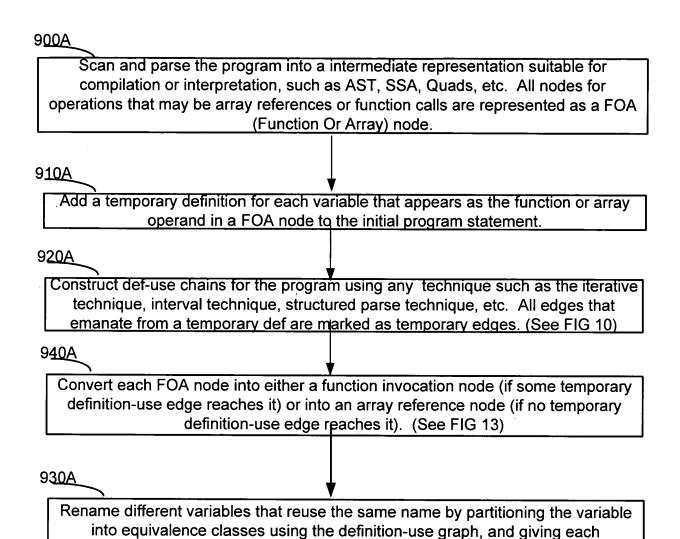


FIG. 9A

equivalence class a name. (See FIG 11).

TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE

4/16/2004 15 / 22

1000—

Compute a control flow graph for the program, dividing the program into basic blocks

1010-

For each statement in the program, compute three lists:

- a) inputs (the variables used by the statement);
- b) outputs (the variables defined by the statement) and
- c) killed (the variables whose definitions are killed by the statement)

1020-

Compute an array *definitions* of the definitions in the program, where each member of an output list is a separate definition;

Set a variable *ndefs* equal to the number of definitions

1030-

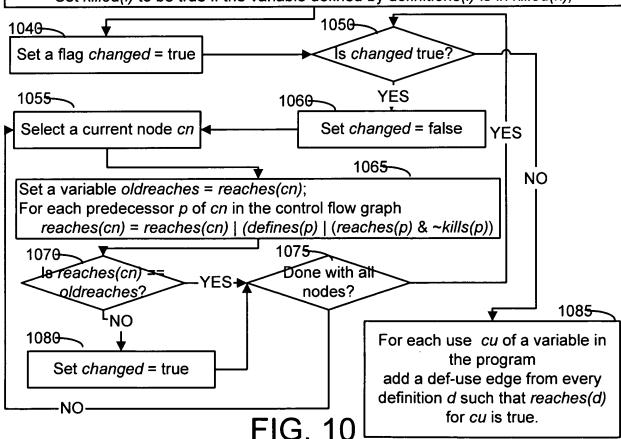
For each node *n* in the control flow graph, allocate 4 bit vectors each of which contains *ndefs* bits, all of which are initially 0:

- a) uses the definitions that are used in that node
- b) defs the definitions that are defined in that node
- c) killed the definitions that are killed in that node
- d) reaches the definitions that reach that node;

Set *uses(i)* to true if the variable defined by *definitions(i)* is in *inputs(n)*;

Set defs(i) to true if the variable defined by definitions(i) is in outputs(n);

Set *killed(i)* to be true if the variable defined by *definitions(i)* is in *killed(n)*;



4/16/2004

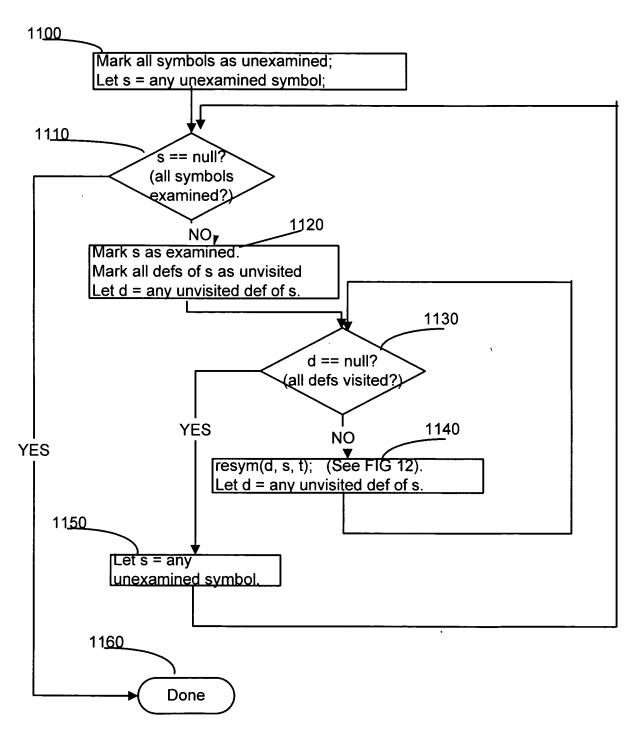


FIG. 11

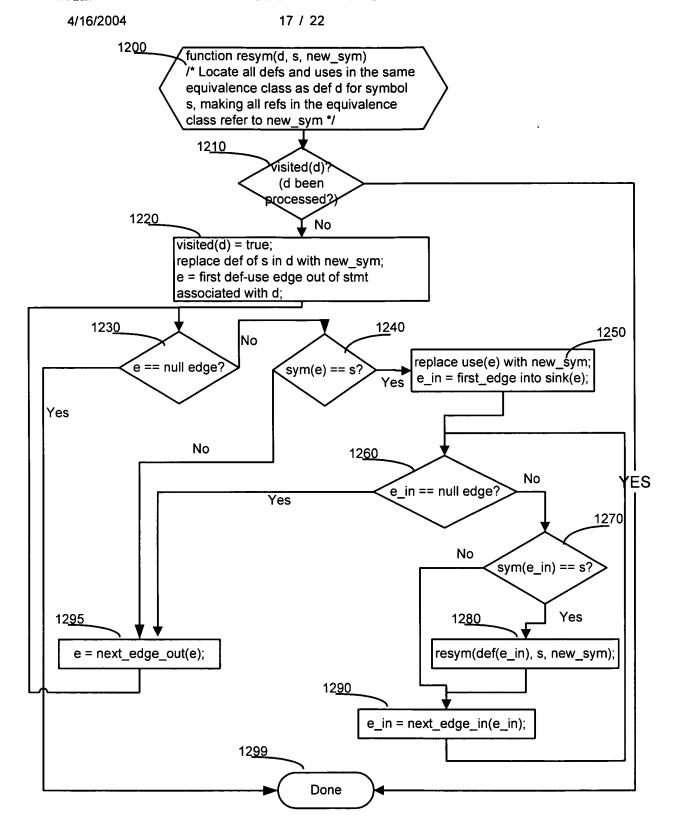


FIG. 12

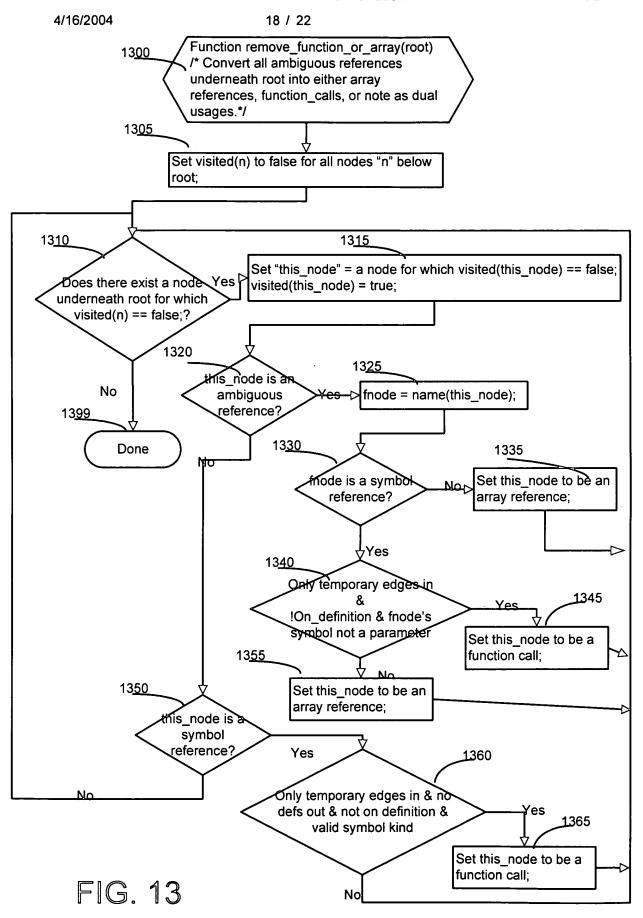
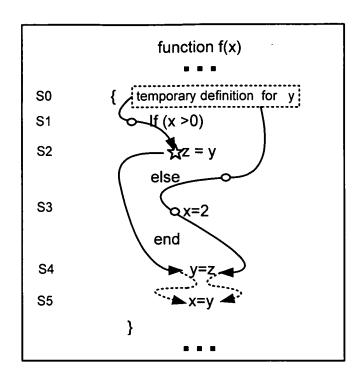


FIG. 14A



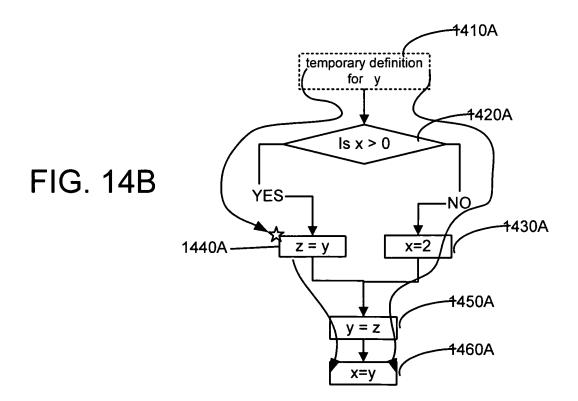
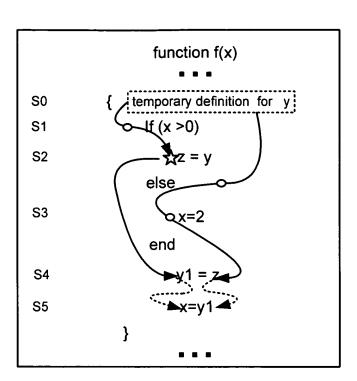
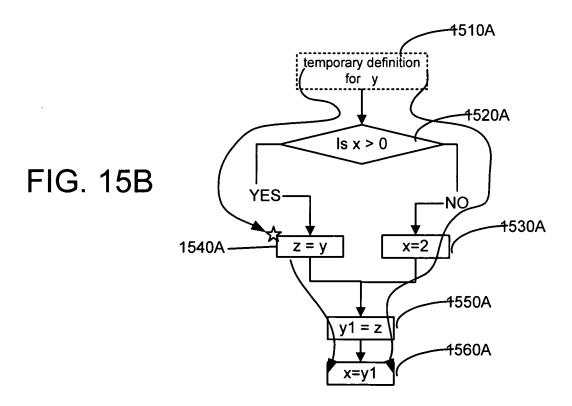


FIG. 15A





4/16/2004 21 / 22

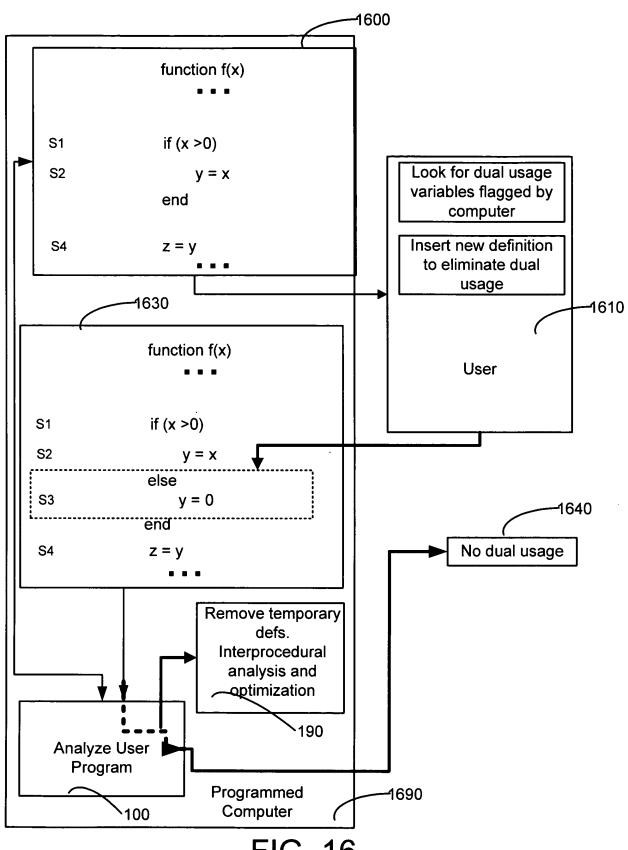


FIG. 16

TITLE: INFERENCING EXISTENCE OF FUNCTION CALLS IN AN AMBIGUOUS LANGUAGE

4/16/2004

22 / 22

FIG. 17A

function f(x)

. . .

S1 [a,b,c] = y(1)

S2 y = x

FIG. 17B

function f(x)

- - -

y(1) = 0

S2 y = x

FIG. 17C

function f(x)

global y

S1 z = y

S2 y = x

FIG. 17D

function f(x)

. . .

S1 z = y

S2 a = y